An additional factor under very heavy loads at present is the overflow of paging traffic to the moving head disks which occurs occasionally as we do not have the more intelligent fixed head storage manager operating yet. When swapping moves to the disks, two things happen. First, the time to move a page in or out increases by a factor of 4 or 5 and second, the disk channel is tied up more often causing contention between swapping and other disk input/output activities. This may cause particularly long load times for very large programs. This latter problem should be remedied by software changes to allocate fixed head space to active pages and to move dormant ones to moving head storage.

From a system design point of view, the remaining response problems are affected by two system resources; memory size and CPU power. Adding more memory would allow more of the runnable jobs to be resident at a given time, thereby giving more continuous attention to more jobs. Increased memory would move the break point in the response curve caused by swapping toward a higher load average. Already though (based on the above comments), at a load average of 2-3 the response is degrading. Since we are running quite efficiently now (15-20% overhead), allowing more jobs to be in core to compete for CPU power would not effectively solve the response problem. It would have the beneficial effect of "linearizing" the degradation at loads above 4.

A second approach would be to increase the inherent processor speed so that jobs would get finished sooner and leave the runnable state more quickly because they are actually completed rather than having their aliquot timed out. This would reduce the load average with the same number of users because jobs would be waiting for user input more of the time. As noted in the report, the experience in upgrading the IMSSS KA-10 to a KI-10 dropped the load average from around 15 to about 5-10. Added CPU power would not change the location of the swapping break on the response vs load average curve, it reduces the load average on the existing curve. It also has the effect of reducing the number of swapping cycles a job has to go through to complete execution when swapping does set in.

In the long term both augmentations would likely be desirable. There is an optimum balance between CPU power and system memory size. That balance is reached for interactive jobs when the memory holds enough runnable jobs that when the processor is distributed over all of these jobs, just acceptable user response is achieved. Additional jobs would cause swapping which incurs a more steeply rising response penalty per added job than if more memory were available, but the response would become unacceptable in either case.

On the basis of current data and since only one of these resources could be upgraded within the present budget, we feel the CPU augmentation is the better choice. It would alleviate the present bottleneck because it improves the user-perceived response time by improving the actual running time of his job. In addition at lower load averages, it would allow more complex programs to become interactive because they would run faster.

For these reasons as the SUMEX-AIM system becomes progressively more heavily loaded with the addition of new users and collaborators, we feel the CPU will be the next most critical resource of the facility. We are currently working on a preliminary plan (see page 51) proposing to allocate available funds within the council-approved award levels to up-grade the present KI-10 CPU. We expect this plan to be refined in the next few months at which time we will submit it for AIM Executive Committee and NIH review.

APPENDIX D

PDP-11 SAIL Design Summary

SAILEX --- SAIL EXperimental compiler

The Portability Problem

The number and availability of computers has expanded greatly in recent years. Accompanying this expansion has been a proliferation of programming languages and software systems. Usually, each computer has its own assembly language, and supports one or more high-level languages. The software is written in assembly language, and hence can execute only on the host computer. The many common functions performed by systems software are often obscured by implementations peculiar to a given computer. Assemblers, compilers, text editors, and linkers require separate manuals for each computer. Languages such as FORTRAN, ALGOL and BASIC have restrictions, extensions, and usually many undocumented idiosyncrasies. Operating systems have widely differing capabilities, though the underlying computers may not be so diverse. Programs often have dependencies on the available environment, or employ peculiarities of a particular computer model.

Such computer-specific programming limits a program's portability, i.e. its ability to execute on more than one computer. Some programs are unlikely candidates for portability, e.g. a program which communicates with a one-of-a-kind device. Many programs, however, could be written so that only minimal changes, if any, would be needed to allow execution on another computer. Since assembly languages are designed for specific computers, portable programs must be written in a higher level language, i.e. a language which does not concern itself with the physical makeup of the computer. Machine-independent languages focus attention on the problem to be solved, rather than the implementation of the solution.

The M x N problem involves M languages which are to be executable on any of N computers. Each language could have a compiler for each machine, resulting in M * N compilers. Alternately, there could be M compilers which translate the languages into a single intermediate code, and N compilers which translate the intermediate code into target code, for a total of M + N compilers. Further reduction in the number of compilers depends on similarities in the languages or target computers. In what follows we shall be concerned with the compilation of a single language for many computers, a reduction of the M x N problem to the 1 x N problem. A distinction should be made between "language portability" and "program portability." By language portability we mean a language which can be compiled into code for many computers. Program portability is the ability of a particular program to be executed on many computers.

Language Portability

A number of new developments in computing will have a great impact on language portability. The growing literature on programming techniques is making programmers more aware of general techniques, and more willing to build on the work of others. Computer networks will lead to large-scale sharing of programs, and the desire to execute programs on one computer which were developed on another. Experience on the various computers available over a network should lead to increased awareness of machine-dependent aspects of a program, and an effort to write programs usable by other computers on the network.

Memory capacity, available peripherals, even instruction sets sometimes vary little from computer to computer. Computer families with differing options between models are becoming commonplace. A single language could be compiled into code tailor-made for a particular model, so that programs written in the language could be independent of the target machine. Internal organizations are often comparable, as evidenced by the common general register --- base/displacement architecture. Increases in speed and memory help alleviate inefficiencies caused by not using a language specifically written for a computer.

Thus the feasibility of and motivation for language portability seem well established. There is a need for easy compilation of programs for different computers. Each compilation should fit the program to the target machine, making as much use as possible of the target instruction set. The compilation should not spend a great deal of time determining the characteristics of the target machine. Decisions concerning code-generation strategy should be made once, then incorporated into the compiler. The development of compilers for new machines should be an integral part of the compiler system, with as much work as possible done automatically. A simple yet flexible manner of specifying target machines should be available. The compiler itself should be written in the language being implemented, so that it can execute on computers for which it can generate code.

Program Portability

Program portability is more difficult to obtain than language portability. A program written in a portable language can be compiled into code for many computers, but execution of the code may produce various results on the different computers. For example, such a program could have dependencies on word size, numeric representation, character representation, memory size, addressability, or the file system. Mathematical routines implemented on computers with differing numeric precision are difficult to port without rather elaborate precautions. Programs which must interact with the operating system, or otherwise "liberate" themselves from the programming language, must be given special attention to minimize such machine-dependence.

Portability should be considered in the initial design of a program. A little inefficiency may be acceptable to make a program portable. The resulting implementation is often more efficient, more flexible, and more easily understood and maintained. When machine-dependence is felt necessary, it should be isolated and well documented as such. Many programs which may seem to be machine-dependent can be made portable by judicious design. For example, text editors often have many dependencies on the operating system and file system. Yet the operating and file systems, and the editors, may have very similar capabilities. Thus the dependencies are not inherent to text editing, and an editor capable of running on all computers could be designed with little loss in efficiency.

SAILEX --- A Machine-independent Compiling System

A machine-independent compiling system is being developed for a subset of the language SAIL, the Stanford Artificial Intelligence Language. SAIL is described in MEMO AIM-204 of the Stanford Artificial Intelligence Laboratory, Stanford University. Changes made to SAIL for use on the TENEX operating system are described in TENEX SAIL, Technical Report No. 248 of IMSSS (Institute for Mathematical Studies in the Social Sciences), Stanford University.

SAIL was originally designed for implementation on a PDP-10 computer, but much of the language is machine-independent. Some parts considered too machine-dependent have been eliminated from SAILEX, and some features have been added to enhance the power of SAIL (e.g. double precision). SAIL is certainly not an ideal language for portable programming, but has some characteristics which make it a suitable experimental language for this purpose. A compiler already exists for SAIL, so that the SAILEX system (which is written in SAIL) can be compiled and executed without hand translation to some other implemented language. The language is powerful enough to give a good test of the feasibility of compiling for many computers. A rather large number of users already exist, and there is great interest in using SAIL on machines other that the PDP-10.

A means of specifying the semantics of a target machine has been designed to meet the previously discussed criteria. This specification includes:

1. External procedures to be available without declaration. For example, double precision numerical routines may be available on some machines.

2. Registers, and register classes. Examples of register classes are integer registers, floating point registers, and index registers.

3. Information needed for storage allocation. For example, the number of addresses per integer.

4. Switches governing operation of the resulting compiler. For example, can the symbol table be kept in memory; is readable intermediate code desired.

5. Code generators. A language has been designed for specifying code generation (the compiler outputs assembly language, not binary code). This language gives the code generators the appearance of the target code being specified. The full power of SAIL can be used to write the code generators, but due to the built-in capabilities of the generator language, such power will probably never be necessary.

The compiler makes almost no assumptions about the target machine. Extensive procedures for manipulating registers and their contents are available, but are used only if invoked by a generator. Much of the work of code generation has been found to be machine-independent; only the specifics of a target machine need be determined. The generators are responsible for requesting loading of registers for instructions which require register addressing, most local optimization, choice of instruction sequence, addressing format, and allocation of storage. The compiler does all the bookkeeping tasks such as remembering what is in the registers; loading, storing, clearing and marking registers as requested; searching for the "optimal" free register; remembering what operands have been used, for later allocation; parameter passing schemes; symbol table maintenance; and file manipulation.

The semantic specification of a target machine is used to create a compiler specifically for that machine. Extensive conditional compilation insures that those parts of the compiler not necessary for a particular machine will not be present. Compilers have been created for the PDP-11/45, PDP-11/40, PDP-10, IBM-SYSTEM/360, VARIAN-620I, and NOVA. More computers are being considered, e.g. CDC family, INTERDATA, BURROUGHS family, DATAPOINT, and SIGMA. The system will continually be generalized as these machines are examined.

Specification of a new machine without radical departures from all previously specified should be straightforward. The PDP-11/40 was specified in two hours, but the specification of the PDP-11/45, which had already been specified, was used as a template. The VARIAN machine took two days, but this included learning the instruction set (the code has not been carefully checked). In general, a poor implementation can be produced very quickly, a compiler can be generated, and the resulting code checked. This indicates how the code can be improved, and the process starts again. A compiler can be completely created from the semantic specification in a matter of minutes (depending, of course, on the speed of the computer being used). A manual describing how to specify a target machine, with extensive examples from those already described, will be produced. Familiarity with SAIL and the specification manual should be

sufficient to produce a compiler for a new target machine with little trouble.

The code produced is not globally optimized, but otherwise rather good. For example, SAILEX appears to produce 20-30% less code for the PDP-10 than the currently available SAIL compiler. Comparison with the code generated by the FORTRAN compiler on the PDP-11 and an "equivalent" SAIL program indicates a reduction in size of about 30%. Such measurements are not precise, and are given only to indicate that SAILEX does not output inefficient code.

Care has been taken to insure that the SAIL compiler is portable. By compiling the compiler, a SAIL compiler can be created which runs on the target machine (a runtime system must be written for the target machine). This has been done for the PDP-11/45, so that the SAIL compiler is available on a PDP-11/45 (also on a PDP-10). (The SAILEX compiler, together with the runtime system and symbol table space, takes about 20K words on the PDP-11/45 (the runtime system is a library). About 2K more words are needed for string space.)

Future areas to be considered are:

1. Final testing of the PDP-11/45 runtime system, and preparation for its export to other sites.

2. Specification of more machines, and resulting generalization of the code generation scheme.

3. Removal of any "hidden" machine dependencies in the compiler system.

4. Implementation of more of SAIL (e.g. full macro facility, records and references).

5. Machine independent global code optimization. The compiler has been designed to facilitate code optimization within a variable size window about the intermediate code. This is not yet done.

6. Design of a machine independent runtime system. The PDP-11/45 runtime system is written in PDP-11/45 assembly language for execution under DOS, version 9. Much of the design could in fact be abstracted from this setting. This abstraction might be viewed as a blueprint from which runtime systems for other machines could be developed, or it might be possible to actually write most of the system in SAIL, and directly export it.

APPENDIX E

Subsystems and Documentation Directories

Nancy Smith
December 1974
(updated April 1975)

The sources of available documentation for these programs will be abbreviated as follows:

TUG	Tenex User's Guide (1975 edition)
DUH	DEC Users Handbook
DAL	DEC Assembly Language Handbook
DML	DEC Mathematical Languages Handbook
HC	a hard-copy manual for the language
OL	on-line documentation which can be found by @DIR <DOC>programname.* . The following extensions are used on the <DOC> directory:

.MANUAL	complete usually fairly long manual
.HELP or .HLP	shorter summary, list of commands, etc.
.SUPPLEMENT	on-line supplement to hard-copy doc
.UPDATE	list of updates by date
.SAMPLE	sample program or output

See <DOC>A-LIST-OF-ALL-AVAILABLE-DOCUMENTS.INFO for complete details on these documents including where and how to order them.

Many of the major programs also have a <BULLETINS>programname.BBD file where messages about new developments, bugs, hints for using the program etc. are sent. These <BULLETINS> files can be read by any of the mail reading programs (READMAIL, RD, or BANANARD).

New programs or new versions of old programs will be put on <NEWSYS> for a trial period. The file <NEWSYS>NEW-SYSTEMS.INFO which is a message file will have a message about each program available. The doc for these new programs will also be kept on the <NEWSYS> directory. These new programs will not be included in the list of programs given here.

The EXEC command @HELP prints a file with general help information.

SUBSYS	DESCRIPTION	DOC
2SIDES	makes files for multi-columns and/or 2-sided listing	OL

```
ACCESS     gives a list of subsys's currently available to GUESTs
ADDMSG     appends a msg to a specified file
AID        algebraic interpretive dialog conversational lang. HC
AIFAIL     assembly lang. - early version of FAIL from SU-AI OL,HC
BAIL       preliminary version of SAIL debugger (on <SAIL>)  OL
BACKUP     short term file loss protection                      OL
BANANARD   msg reading program (many extra features)              OL
BASIC      conversational programming lang. (DEC  version) OL,DML,TUG
BCPL       compiler writing and systems programming lang.         HC
BINCOM     binary comparison of files (now replaced by FILCOM) DAL
BLIS10     compiler for system implementation (DEC version) OL,HC,TUG
BLIS11     BLISS for the PDP11
BLISS      compiler for system implementation (TENEXized)  OL,HC(DEC)
BUDGET     budget management program (especially proposals)  OL
BYE        @BYE same as @BREAK (LINKS)
CALENDAR   calendar management and reminder system           OL,TUG
CAM        the compare and merge program of SOUP see <DOC>SOUP.MANUAL
CCL        concise command language                             OL,DUH
COPYM      reading/writing DECtapes                             OL,TUG
CREF       cross-reference assembly listing                     OL,DAL
CRSREF     TENEX cross-referencing program (outfile_infile(s))
DCHANGE    character set conversion for "foreign" tapes         OL
DCHECK     reads blocks of file into core & calls DDT to examine OL
DDT        debugger (single-stepping added at IMSSS)      OL,TUG,DAL
DED        text-editor (designed for TENEX)                     OL
DELOLD     deletes files by cutoff date of last access          OL
DELVER     deletes excess versions of files                     TUG
DIABLO     prints final copy of PUB-produced documents on DIABLO  OL
DIRNUM     translates directory name to number for DEC programs OL
DO         creates or appends a line to a reminder file         OL
DONE       deletes a line from a reminder file                  OL
DROP       similar to DELVER, deletes oldest and 2nd newest on *.*
DTACOP     DECtape to DECtape copy
DUMPER     reads/writes magnetic tapes
EE         @EE <program>  runs program on your directory as ephemeral
ES         @ES <program>  runs program on <SUBSYS> as ephemeral
EXTR       "EXTRactor" processes MACRO/FAIL source files to
             produce .FAI listing of labels defined
F40        FORTRAN IV (see also <DOC>FORTRAN.HELP and      OL,TUG,DML
             <DOC>LISP-FORTRAN-INTERFACE.HELP )
FAIL       assembly language (BBN version of FAIL)              OL,HC
           e also JSYS manual & <DOC>JSYS.INFO)
FED        the final edit program of SOUP see <DOC>SOUP.MANUAL
FILCHK     checks SAIL programs for  loader incompatibilities   OL
FILCOM     complete file comparison package              OL,DAL,TUG
FILDMP     dumps files in variety of formats                       OL
FILES      multiple to multiple copies, renames, protections
FORTRA     FORTRAN10(version 1A) (see also <DOC>FORTRAN.HELP)  OL,HC
FREQ       ranks words in text file according to frequency
FRKCOM     compares an address space with address space of file TUG
FTP        ARPANET file transfers                                  TUG
FUDGE2     updates/manipulates files containing rel programs  DAL,TUG
GETDMP     loads into core .dmp file from SU-AI (SAV only to
             677777) type filename to * prompt
GRIPE      sends comments or complaints about system to staff   TUG
HELP       prints out short general help file for SUMEX or help for
```

		programs
HOSTAT	prints network site status information	TUG
IFAIL	assembly language (IMSSS version of FAIL)	OL,HC
ILISP	UC Irvine LISP (extension of LISP 1.6)	OL
IMSSS	direct link to IMSSS	
LD	prints SYSTAT-like info including msg if MAIL WAITING	
LINK10	DEC loader	OL,DAL,TUG
LINK11	linker for PDP11 DOS operating system	
LINKSTAT	prints status of IMSSS link	
LISP	INTERLISP-see also <DOC>LISP-FORTRAN-INTERFACE.INFO	OL,HC
LOADER	(from IMSSS)-see <DOC>LINK10-LOADER-DIFFERENCES.HELP	TUG
LOADGT	GT40 standard format loader	
MACRO	assembly lang.-see JSYS manual & <DOC>JSYS.INFO	TUG,DAL
MAILSTAT	info on queued mail	TUG
MANTIS	Fortran debugger	
MLAB	mathematical modeling and graphics package	OL
MULTI	multiple-fork supervisor--switches between forks	
MY-ACCOUNTS	prints user's valid accountnames	
NETSTAT	prints info on ARPANET status	TUG
NON	zero-compresses file, options to remove linenumbers, pagemarks, etc.	
PCSAMP	measures the operation of other user programs	TUG
PIP	DEC utilities program	OL,DUH
PIP11	transfers PDP11 DOS DECtapes to/from TENEX files	OL
PNTMAK	converts underlines to suitable format for LPT:	OL
POET	text editor designed for TENEX use	OL
PPL	an interactive extensible programming lang.	TUG
PROFIL	gives freq of execution of SAIL statements	OL,HC
PUB	document preparation lang.	OL
RD	mail reading program (BANANARD is better)	TUG
READMAIL	mail reading program " "	TUG
RECORD	for pseudo-ttys, typescript of job, detaching from running job	OL
REDUCE	symbolic algebraic language	OL
RUNFIL	uses file instead of tty for input commands	TUG
RUNOFF	document-preparation language (DEC not BBN version)	OL
SAIL	ALGOL-like lang.-see also <DOC>LEAP.MANUAL	OL,HC
SCAN	scans multi-directories for a variety of file info	OL
SEARCH	searches multi-text files for English words or SAIL identifiers, can be used with TV editor	OL
SEGSAV	reads .shr & .low files to produce TENEX .sav	OL
SITBOL	compiler version of SNOBOL	OL
SNDMSG	message sender	OL,TUG
SNOBOL	string-processing programming lang.	OL,HC
SORT	stand alone COBOL column-oriented text file sorter	OL,TUG
SOS	text editor	OL
SPELL	spelling checker/corrector for text files (not TENEX)	OL
SRCCOM	compares text files	TUG
SWITCH	switches the format of a reminder file	OL
SYSIN	executes LISP SYSOUT's	OL
TABLE	creates conversion tables for DCHANGE	
TALK	used with LINK command to eliminate need for ;'s	
TAPCNV	reads card image file processed by MTACPY	TUG
TBASIC	TENEXized version of DARTMOUTH BASIC	OL
TCTALK	teleconferencing over ARPANET	OL

TECO	text editor (see TENEX TECO manual)	OL,TUG
TMERGE	merges specified text pages from files into new file	OL
TODAY	lists the contents of today's reminder file	OL
TRITAP	processes magtapes from XEROX, IMSSS, BBN	OL
TTYTRB	used to report terminal line problems	TUG
TTYTST	prints test patterns for diagnosing terminal	TUG
TV	text editor for TEC and DATAMEDIA displays	OL
TVFIX	restores bad TV files (see <DOC>TV.MANUAL)	
TYMSTAT	(for TYMNET lines only) gives measure of current efficiency of TYMNET transmission	
TYPBIN	does an octal dump of a packed file	TUG
TYPREL	analyzes contents of .REL files	TUG
WATCH	continuous on-line monitoring of system activity	TUG
WATCH.IMS	IMSSS version of WATCH	
WHAT	lists the contents of a reminder file	OL
WHO	prints SYSTAT-like information	
WHOIS	looks up username & prints name/address info on user	OL
XED	text-editor (used with BANANARD)	OL
XT	reformats and prints text file	OL
Z	logs jobs off from inferior forks	

<DOC> DIRECTORY LISTING -

The following is a listing of the <DOC> directory which contains most of the on-line formal documentation about the system and subsystem.

```
   <DOC>    18-MAY-75 16:06:22

 2SIDES.HELP;1
 A-GENERAL.HELP;11
 A-GUIDE-TO-TENEX-USER'S-GUIDE.INFO;1
 A-LIST-OF-AVAILABLE-DOCUMENTATION.INFO;7
 A-SURVEY-OF-THE-DEC-HANDBOOKS.INFO;9
 ACCOUNT-NAME-USAGE.INFO;1
 ALL-SUBSYS'S-AVAILABLE-AT-SUMEX.INFO;4
 BACKUP.HELP;1
 BAIL.HELP;2
 BANANARD.MANUAL;2
    .UPDATE;3
 BASIC.HLP;1
 BBN-PROGRAM-VERSION-NUMBER-STANDARDS.INFO;1
 BLIS10.HLP;2
 BLISS.HELP;1
 BSYS.MANUAL;2
 BUDGET.SAMPLE;2,1
    .MANUAL;5,4,3
 CALENDAR.MANUAL;1
 CCL.HELP;4
    .UPDATE;1
 CHECKDSK.HELP;2
 CHESS.HELP;1
 CLEAN.HELP;1
 COPYM.HELP;2
 CREF.HLP;1
    .UPDATE;1
 DCHANG.HLP;1
 DCHANGE.MANUAL;1
 DCHECK.HELP;1
 DDT.SUPPLEMENT;1
 DEC-HANDBOOK-GLOSSARY-UPDATE.INFO;1
 DEC/TENEX-COMMAND-EQUIVALENTS.INFO;1
 DED.MANUAL;1
 DELOLD.HELP;1
 DESCRIPTION-OF-SUMEX-AIM-PROJECTS.INFO;3
 DIABLO.HELP;3
 DIRNUM.HELP;1
 DO .HELP;4
 DUMP.INFO;1
 EDIT.INFO;1
 EDITOR-PROGRAM-INTERFACE.INFO;1
 FAIL.MANUAL;1
    .HELP;4
```

```
FILCHK.HELP;1
FILCOM.HLP;4
FILDMP.HELP;2
FILEX.UPDATE;1
   .DOC;1
FORDDT.DOC;1
   .HLP;1
FORTRA.HLP;1
FORTRAN.HELP;2
HOW-TO-UPDATE-DOC.INFO;3
IDDT.HELP;1
ILISP.MANUAL;1
INTERROGATE.HELP;4
INTRO-TO-SUMEX-AIM-TENEX.INFO;4
LEAP.MANUAL;1
LINK10.HLP;1
   .DOC;1
LINK10-LOADER-DIFFERENCES.HELP;1
LISP.HELP;3
   .UPDATE;1
LISP-FORTRAN-INTERFACE.HELP;2
LIST.HELP;3
MACRO.HLP;1
   .DOC;1
MINI-DUMP.LISTING;1
MLAB.HLP;2
NAIVE.PUB;1
NSOS.SUPPLEMENT;1
   .MANUAL;1
   .INTRO;1
OVERVIEW-OF-COMPUTER-SYSTEM.INFO;1
PIP.HLP;1
   .SUPPLEMENT;1
   .UPDATE;1
PIP11.HELP;1
PNTMAK.HELP;1
POET.HELP;1
   .MANUAL;1
PROFIL.UPDATE;2
PROJECTS-AND-ASSOCIATED-USERS.INFO;8
PUB.MANUAL;2
   .HELP;4
   .UPDATE;9
PUB-MANUAL.PUB;10
RECORD.MANUAL;1
REDUCE.MANUAL;1
RUNOFF.MANUAL;2
SAIL.HELP;1
   .SUPPLEMENT;2
   .UPDATE;3
   .TENEX-SUPPLEMENT;1
   .BEGIN-MANUAL;1
SAMPLE.PUB;1
SCAN.HELP;1
SEARCH.MANUAL;3
   .INFO;3
```

```
SEGSAV.HELP;1
SETTING-UP-NEW-USER-DIRECTORIES.INFO;1
SITBOL.HELP;1
SNDMSG.HELP;6
SNOBOL.MANUAL;1
SORT.HLP;1
SOS.UPDATE;6
   .HELP;4
   .MANUAL;1
SOUP.HLP;1
   .MANUAL;1
SPELL.MANUAL;9
SUMEX-JSYS'S.INFO;1
SYSIN.HELP;1
SYSTEM-SCHEDULE.INFO;3
TBASIC.HELP;2
   .SAMPLE;1
   .MANUAL;4
TCTALK.DOC;1
TECO.SAMPLE;1
   .COMMANDS;1
   .HELP;1
   .TEXT1;1
   .TEXT2;1
   .SUMMARY;1
TELNET.INFO;1
TENEX-EXEC-MANUAL-UPDATE.INFO;5
TERMINAL-LINKING.INFO;1
TMERGE.HELP;3
TRITAP.HELP;1
TV .UPDATE;8
   .MANUAL;6
TV-STRINGS.PMAP;1
TYMNET-INSTRUCTIONS.INFO;1
USER-NAME-ADDRESS-PHONE.INFO;29
WHOIS.HELP;1
XED.HLP;1
   .MANUAL;1
XT .HELP;1
 142 FILES, 1332 PAGES
```

<BULLETINS> DIRECTORY LISTING -

The following is a listing of the <BULLETINS> directory which is a repository of informal or transient information about the system, subsystems, current events, and items of interest.

```
   <BULLETINS>    18-MAY-75 16:06:42

 12-MAR-75.SYSLTR;1
 ARPANET.BBD;1
 ASCII.BBD;1
 BASIC.BBD;1
 BULLETINS.BBD;1
 CALENDAR.BBD;4,3,1
 COMPATIBILITY.BBD;1
 CONSTANTS(PHYSICAL-OR-CHEMICAL).BBD;1
 DIURNAL-LOADING-WEEKDAYS.MAR;1
    .3/31/75;1
    .2/17/75;1
    .2/24/75;1
    .3/3/75;1
    .3/10/75;1
    .3/17/75;1
    .3/24/75;1
    .APR;1
    .FEB;1
 DO .BBD;1
 EDIT.BBD;1
 EMPLOYMENT-WANTED.BBD;1
 FILES.BBD;1
 FORTRAN.BBD;1
 GOOD-LISP-USAGE.BBD;2
 GOOD-SYSTEM-USAGE.BBD;1
 GUEST-LIST.BBD;2
 IN-WATS.BBD;1
 KWIC.BBD;1
 LIBRARY-SAIL.BBD;1
 LINK10.BBD;1
 LINKING.BBD;1
 LISP.BBD;2
 LIST.BBD;1
 LOGIN-CMD.BBD;1
 LOGIN-MESSAGES.BBD;2
 MACRO.BBD;1
 MEETINGS.BBD;1
 MLAB.BBD;1
 NEW-EXEC.INFO;1
 OLD-SYSTEM-MESSAGES.BBD;1
 PDP11-GT40.BBD;1
 POSITIONS-AVAILABLE.BBD;1
 PROTECTION.BBD;1
 RECORD.BBD;1
```

```
SAIL.BBD;1
   .   ;1
SEARCH.BBD;1
SNDMSG-READMAIL.BBD;1
SORT.BBD;1
SOS.BBD;1
SPELL.BBD;1
SYSTEM-MESSAGES.BBD;1
TECO.BBD;1
TENEX.BBD;1
TESTIMONIALS.BBD;1
TV .BBD;1
TV-STRINGS.PMAP;1
TYMNET.BBD;1
 60 FILES, 149 PAGES
```

APPENDIX F

Networking and Collaborative Research - DENDRAL Project

The following is a preprint of a paper to be presented at the 170th meeting of the American Chemical Society in Chicago during August 1975 - the symposium title is "Computer Networking in Chemistry". This paper will appear in the proceedings and reflects well the orientation and activities of the SUMEX-AIM resource and its collaborating projects (DENDRAL in this case).

NETWORKING AND A COLLABORATIVE RESEARCH COMMUNITY: A CASE STUDY USING THE DENDRAL PROGRAMS.

Raymond E. Carhart*, Suzanne M. Johnson, Dennis H. Smith, Bruce G. Buchanan, R. Geoffrey Dromey, and Joshua Lederberg.

Departments of *Computer Science, Genetics, and Chemistry, Stanford University, Stanford, California, 94305.

Computer Science is one of the newest, but also one of the least "cumulative" of the sciences. Gordon (1) has recently pointed out the upsetting disparity between the number of potentially sharable programs in existence and the number which are easily accessible to a given researcher. Although some mechanisms exist for the systematic exchange of program resources, for example the World List of Crystallographic Computer Programs (2), a great deal of programming effort is duplicated among different research groups with common interests. The reasons for this are understandable: these groups are separated by geography, by incompatibilities in computer facilities and by a lack of a means to keep abreast of a a rapidly changing field.

The emergence of more economical technologies for data communications provides, in principle, a method for lowering these geographical and operational barriers; for creating, through computer networking, remote sites at which functionally specialized capabilities are concentrated. The SUMEX-AIM (Stanford University Medical EXperimental computer - Artificial Intelligence in Medicine) project is an experiment in reducing this principle to practice, in the specific area of artificial intelligence research applied to health sciences.

The SUMEX-AIM computer facility (3) is a National Shared Computing Resource being developed and operated by Stanford University, in partnership with and with financial support from the Biotechnology Resources Branch of the the Division of Research

Resources, National Institutes of Health. It is national in scope in that a major portion of its computing capacity is being made available to authorized research groups throughout the country by means of communications networks.

Aside from demonstrating, on managerial, administrative and technical levels, that such a national computing resource is a viable concept, the primary objective of SUMEX-AIM is the building of a collaborative research community. The aim is to encourage individual participants not only to investigate applications of artificial intelligence in health science, but also to share their programs and discuss their ideas with other researchers. This places a responsibility upon SUMEX-AIM to develop effective means of communication among community members and among the programs they write. It also places responsibility upon those members to design and document programs that readily can be used and understood by others.

Another aspect of the SUMEX facility is providing service to individuals whose interest is in using, rather than developing, the available computer programs. Although this is not a primary consideration, it is an essential part of the growth of these programs. Most of the SUMEX-AIM projects have formed, or are forming, their own user communities which provide valuable "real world" experience. Figure 1 depicts the typical interaction of such a project with its user community, and with other projects. The participation by users in program development is not just restricted to suggestions, but can also include software created by computer-oriented users to satisfy special needs. In some projects, methods are being considered to further promote this kind of participation.

The purposes of this paper are threefold: first, to indicate the range of research projects currently active at SUMEX; second, to describe in detail one of these projects, DENDRAL, which is of particular interest to chemists; and third, to discuss some problems and possible solutions related to networking and community-building.

I. Research Activities at SUMEX-AIM

The community of participants in SUMEX-AIM can be divided geographically into local (i. e., Stanford-based) projects and remote projects, and below is given a brief description of the major representatives of each. Communication with the remote projects is accomplished through one or both of the communications networks shown in Figure 2. In most cases, connection with SUMEX-AIM from these remote sites involves only a local telephone call to the nearest network "node".

The SUMEX-AIM system is itself undergoing constant improvement which deserves to be called research, and thus a third section is included here to represent system developments.

Remote projects

The Rutgers project. Originating from Rutgers University are several research efforts designed to introduce advanced methods in computer science - particularly in artificial intelligence and interactive data base systems - into specific areas of biomedical research. One such effort involves the development of computer-based consultation systems for diseases of the eye, specifically the establishment of a national network of collaborators for diagnosis and recommendations for treatment of glaucoma by computer. Another project concerns the BELIEVER program, which represents a theory of how people arrive at an interpretation of the social actions of others. SUMEX-AIM provides an excellent medium for collaboration in the development and testing of this theory. The Rutgers project includes, in addition, several fundamental studies in artificial intelligence and system design, which provide much of the support needed for the development of such complex systems.

The DIALOG project. The DIAgnostic LOGic project, based at the University of Pittsburgh, is a large scale, computerized medical diagnostic system that makes use of the methods and structures of artificial intelligence. Unlike most other computer diagnostic programs, which are oriented to differential diagnosis in a rather limited area, the DIALOG system has been designed to deal with the general problem of diagnosis in internal medicine and currently accesses a medical data base which encompasses approximately fifty percent of the major diseases in internal medicine.

The MISL Project. The Medical Information Systems Laboratory at the University of Illinois (Chicago Circle campus) has been established to explore inferential relationships between analytic data and the natural history of selected eye diseases, both in treated and untreated forms. This project will utilize the SUMEX-AIM resource to build a data base which could then be used as a test bed for the development of clinical decision support algorithms.

Distributed Data-Base System for Chronic Diseases. This project, based at the University of Hawaii, seeks to use the SUMEX-AIM facility to establish a resource sharing project for the development of computer systems for consultation and research, and to make these systems available to clinical facilities from a set of distributed data bases. The radio and satellite links which compose the communication network known as the ALOHANET, in conjunction with the ARPANET, will make these programs available to other Hawaiian islands and to remote areas of the Pacific basin. This project could well have a significantly beneficial effect on the quality of health care delivery in these locations.

Modeling of Higher Mental Functions. A project at the University of California at Los angeles is using the SUMEX-AIM facility to construct, test, and validate an improved version of the computer simulation of paranoid processes which has been developed. These simulations have clinical implications for the understanding, treatment, and prevention of paranoid disorders. The current interactive version (PARRY) has been running on SUMEX-AIM and has

provided a basis for improvement of the future version's language recognition capability.

Local Projects

The Protein Crystallography Project. The Protein Crystallography project involves scientists at two different universities (Stanford and the University of California at San Diego), pooling their respective talents in protein crystallography and computer science, and using the SUMEX-AIM facility as the central repository for programs, data and other information of common interest. The general objective of the project is to apply problem solving techniques, which have emerged from artificial intelligence research, to the well known "phase problem" of x-ray crystallography, in order to determine the three-dimensional structures of proteins. The work is intended to be of practical as well as theoretical value to both computer science (particularly artificial intelligence research) and protein crystallography.

The MYCIN project. MYCIN is an evolving computer program that has been developed to assist physician nonspecialists with the selection of therapy for patients with bacterial infections. The project has involved both physicians, with expertise in the clinical pharmacology of bacterial infections, and computer scientists, with interests in artificial intelligence and medical computing. The MYCIN program attempts to model the decision processes of the medical experts. It consists of three closely integrated components: the Consultation System asks questions, makes conclusions, and gives advice; the Explanation System answers questions from the user to justify the program's advice and explain its methods; and the Rule-Acquisition System permits the user to teach the system new decision rules, or to alter pre-existing rules that are judged to be inadequate or incorrect.

The DENDRAL project. This project, being of particular chemical interest, is described in detail in Section II. Through the SUMEX-AIM facility DENDRAL has gained a growing community of production-level users whose experience with the programs is a valuable guide to further development. Although technically users, some members of this community might better be described as collaborators because they have provided SUMEX-AIM with various special-purpose programs which are of interest to other chemists and which extend the usefulness of the DENDRAL programs.

SUMEX-AIM System Development

Current research activities at SUMEX-AIM are developing along several lines. On a system development level there are ongoing projects designed to make the system more user oriented. Currently, the system can be expected to provide help to the user who is confused about what is expected in response to a certain prompt. A "?" typed by the user, will, in most cases, provide a list of possible responses

from which to choose. Also available in response to typing "HELP" to the monitor is a general help description containing pointers to files likely to be of interest to a new user.

In an effort to facilitate communication between collaborators, a program called CONFER has been developed to provide an orderly method for multiple participant teletype "conference calls". Basically, the program acts as a character processor for all the terminals linked in the conference, accepting input from only one at a time, and passing it out to the remaining terminals. In this way, the conference, in effect, has a "moderator" terminal, thus allowing for a more orderly transfer of ideas and information.

SUMEX-AIM is also aware of the necessity of making its facilities available for trial use by potential users and collaborators. To this end, a GUEST mechanism has been established for persons who wish to have brief, trial access to certain programs they feel may be of value to them, and about which they would like to obtain more knowledge. This provides a convenient mechanism whereby persons, who have been given an appropriate phone number and LOGIN procedure, can dial up SUMEX-AIM and receive actual experience using a program they may only have heard about.

Another area of system development currently being explored at SUMEX-AIM is that of creating a comprehensive "bulletin board" facility where users can file "bulletins", that is, messages of interest to the SUMEX-AIM community. The facility will also alert users to new bulletins which are likely to be of interest to them, as determined by individual user-interest profile.

II. DENDRAL - CHEMICAL APPLICATIONS OF INTERACTIVE COMPUTING IN A NETWORK ENVIRONMENT

The major research interest of the DENDRAL Project at Stanford University is application of artificial intelligence techniques for chemical inference, focusing in particular on molecular structure elucidation. Portions of our research are in the area of combined gas chromatography/high resolution mass spectrometry and include instrumentation and data acquisition hardware and software development. This area is beyond the scope of this report; we focus instead on the concurrent development of programs to assist chemists in various phases of structure elucidation beyond the point of initial data collection. SUMEX-AIM provides the computer support for development and application of these programs.

Another aspect of our research is our commitment to share developments among a wider community. We feel that several of our programs are advanced enough to be useful to chemists engaged in related work in mass spectrometry and structure elucidation in general. These programs are written primarily in the programming language INTERLISP, and thus are not easily exportable (exceptions are

indicated subsequently). SUMEX-AIM provides a mechanism for allowing others access to the programs without the requirement for any special programming or computer system expertise. The availability of the SUMEX facility over nationwide networks allows remote users to access the programs, in many instances via a local telephone call.

Much of the following discussion is preliminary because our programs have only recently been released for outside use. Some announcement of their availability has been made, and other announcements will occur in the near future, through talks, publications in press, demonstrations and informal discussions. Although most of our experience has been with local users, they have been good models of remote users in that their previous exposure to the actual programs and computer systems is minimal. Their experience has been extremely useful in helping us to smooth out clumsy interactions with programs and to locate and fix program bugs. Such polishing is important for programs which may be utilized by users from widely differing backgrounds with respect to computers, networks and time sharing systems. We are in the processes of building a community of remote users. We actively encourage such use for two reasons: 1) we feel the programs are capable of assisting others in solving certain molecular structure problems, and 2) such experience with outside users will be a tremendous assistance in increasing the power of our programs as the programs are forced to confront new real-world problems.

The remainder of this section outlines the programs which are available via SUMEX, the utilization of these programs in helping to solve structure elucidation problems and the limitations we see to their use. We discuss current applications of the programs to our research and the research of other users to illustrate better the variety of potential applications and to stimulate an interchange of ideas. Where appropriate, we point out current difficulties with the use both of our programs and of SUMEX. New applications and wider use will certainly change the nature of these problems; we strive to solve current problems, but new ones will always arise to take their place.

DENDRAL Programs

We have several programs which we employ in dealing with various aspects of problems involving unknown structures. Some of these programs are exportable, while the remainder are available at SUMEX. The availability of each program is discussed below.

Our initial emphasis in studying applications of artificial intelligence for chemical inference was in the area of mass spectrometry(4-6). This emphasis remains because many of our problems require mass spectrometry as the analytical tool of choice in providing structural information on small quantities of sample. More recently, we have been developing a program (CONGEN, below) directed at more general aspects of structure elucidation. This has extended the scope of problems for which we can provide computer assistance.

We will begin, however, with discussion of the mass spectrometry

programs. The examples used in the discussion are characteristic of our current research problems, although we have focused on relatively simple problems to keep the presentation brief. We trace, in what might be chronological terms, the application of the programs to various phases of a structure problem. In this way we hope to illustrate the place of each program in the analysis. We begin by discussing preprocessing of mass spectral data (CLEANUP and MOLION). Subsequent analysis of such data in terms of structure is then covered (PLANNER). The use of CONGEN is discussed for problems which cannot be handled by the previous programs. Finally, we discuss efforts to discover, with the use of the computer, systematics in the behavior of known substances in the mass spectrometer as a means of extending the knowledge of the system for applications in new areas (INTSUM and RULEGEN).

Programs for Molecular Structure Problems

The first three programs, CLEANUP, the library-search program and MOLION are in a sense utility programs, but all three play a critical role in processing mass spectral data. Subsequent applications of programs (e.g., PLANNER) for more detailed spectral analysis in terms of structure depend on the successful treatment of the data by CLEANUP and MOLION, while the library search program filters out common spectra which need not undergo a full analysis. The examples used are drawn from our collaboration with persons in the Genetics Research Center at Stanford Hospital. The experimental data which are collected are the results of combined gas chromatographic/low resolution mass spectral (GC/LRMS) analysis of organic components (chemically fractionated and derivatized where necessary) of body fluids, e.g. blood, urine. A typical experiment consists of 500-600 individual mass spectra for each fraction, taken sequentially over time as the various components, largely separated from one another, elute from the gas chromatograph and pass into the mass spectrometer. Each mass spectrum consists of the mass analyzed fragment ions of the component(s) in the mass spectrometer at the time the spectrum was taken. Such spectra are related, indirectly, to the molecular structure of the component(s).

CLEANUP(7). The individual mass spectra obtained from fractionated GC/LRMS analysis are quite often poor representations of corresponding spectra taken from pure compounds. They can be contaminated by the presence of additional peaks and/or distortions of the intensities of existing peaks in the spectrum. Fragment ions from either the liquid phase of the GC column or from components incompletely separated by the gas chromatograph are are responsible for the contamination. We have developed a program, referred to here as CLEANUP, which examines all mass spectra in a GC/LRMS run, selects those spectra which contain ions other than background impurities, and remove contributions from background and overlapping components. A spectrum results which compares favorably with the spectrum of a pure component. Biller and Biemann (8) have developed a similar but less powerful program.

For example, the CLEANUP program detected components at points marked with a vertical bar in the (partial) plot of total ion current vs. scan number (time), Figure 3. Note that overlapping components were detected under the envelopes of the GC peaks in the region of scans 485-488, 525-529 and 539-552. We focus our attention on the spectrum recorded at scan 492. The raw data, prior to cleanup, are presented in Figure 4 (top). The spectrum resulting from CLEANUP is presented in Figure 4 (bottom). Note that the large ions (e.g., m/e 207, 221 and 315) from background impurities are removed, and that the intensity ratios of peaks at lower masses (e.g., 51 and 77) have been adjusted to reflect their true intensities in the spectrum.

The CLEANUP program is capable of detection of quite low-level components in complex mixtures as indicated by some of the areas of the total ion current plot (Figure 3) where components were detected. It is completely general because nothing in the program code is sensitive to the types of compounds analyzed or the characteristics of possible impurities associated with the compounds or from the GC column. Its major limitation is that mass spectra must be taken repetitively during the course of a GC/MS run. Its performance is enhanced when such spectra are measured closely in time.

The program is offered via SUMEX as an adjunct to use of our other programs; it is not offered as a routine service. Because the program is written in FORTRAN, we routinely use it on our data acquisition computer system so as not to burden SUMEX with tasks better done elsewhere. Similarly, we would assist other frequent users to mount the program on their own systems.

Library Search. With a set of "clean" mass spectra available, the next problem is identification of the various components. Over the course of several years, libraries of mass spectral data have been assembled(9). These libraries can be very useful in weeding out from a group of spectra those which represent known compounds(10). Clearly, one should spend time on solving the structures of unknown compounds, not on rediscovering old ones. The CLEANUP program provides mass spectra which are of sufficient quality to expect that known compounds would be identified easily from such libraries.

The spectra detected by CLEANUP in the region of scans 480 to 580 (Figure 3) were matched against the library of biomedically relevant spectra compiled by S. Markey (National Institutes of Health) and our extensions to that library (we wish to thank S. Grotch, Jet Propulsion Laboratory, Pasadena, Ca. for providing some of the library matchings). Excellent matches with the library were obtained for scans 492, 496, 509, 519, 529 and 548. The components are indole acetic acid methyl ester, di-n-butylphthalate, caffeine, salicyluric acid methyl ester, methoxyhippuric acid methyl ester and n-C24 hydrocarbon respectively (structures given in Figures 3 and 4). Spectra scans at 485, 487, 525, 530, 536, 539, 554 and 576 did not match well with any spectrum in the library and thus must be examined further for structural information. The necessity for preprocessing the data using CLEANUP prior to library matching is illustrated from indole acetic acid methyl ester (scan 492). The "clean" spectrum

(Figure 4, bottom) was matched to the library spectrum of this compound much better than to that of any other compound. The raw spectrum (Figure 4, top), when compared to the library resulted in eleven other compounds which matched more closely than the correct one.

This brief example illustrates the obvious value and limitations of library searching. The most interesting compounds for subsequent analysis are those which are unknown. The fractions of urine extracts are replete with unidentified compounds because of the inadequacy of current library compilations. As new compounds are identified they are, of course, added to the library, so that future analyses need not reinvestigate the same material.

We currently perform library searching on our data acquisition and reduction computer systems. We can, if necessary, offer limited library search facilities via SUMEX. However, because commercial facilities are available (e.g., over the GE network), routine library search service is not available on SUMEX.

MOLION(11). At this stage we are left with a collection of mass spectra of unknown compounds. The library search results may have provided some clues as to the type of compound present, e.g., compound class. Structure elucidation now begins in earnest. The key elements in problems of structure elucidation are the molecular weight and empirical formula of a compound. Without these essential data, the structural possibilities are usually too immense to proceed further. Mass spectrometry is frequently used to determine molecular weights and formulae, but there is no guarantee that the mass spectrum of a compound displays an ion corresponding to the intact molecule. For example, many of the derivatives of the amino acid fractions of urine display no molecular ions. When we are given only the mass spectrum (and for GC/MS analysis a mass spectrum may be all that is available) we must somehow predict likely molecular ion candidates. The program MOLION performs this task. Given a mass spectrum, it predicts and ranks likely molecular ion candidates independent of the presence or absence of an ion in the spectrum corresponding to the intact molecule. The published manuscript(11) provides many examples of the performance of the program.

The mass spectrum of an example, unknown X, (which we will pursue in more detail below) is given in Figure 5. The results obtained from MOLION are summarized in Table I. The observed ion at m/e 263 is ranked as the most likely candidate.